BAB II

TINJAUAN PUSTAKA DAN LANDASAN TEORI

BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Tinjauan Pustaka

Penelitian relevan yang dilakukan oleh [2] adalah mengotomasi sistem pendukung keputusan penerimaan peserta ujian mandiri Universitas Negeri Jakarta. Tujuan dari penelitian ini adalah membuat sistem yang terotomatisasi dalam menghitung nilai akhir peserta ujian mandiri dan membantu dalam menentukan diterimanya atau tidak diterimanya seorang peserta dalam program studi yang dipilih sesuai kuota program studi tersebut. Metode pengembangan perangkat lunak yang dipakai adalah metode *Waterfall*. Dengan kriteria yang digunakan yaitu nilai komponen uji program studi untuk kemampuan IPA/IPS, kemampuan dasar, dan juga nilai ketrampilan tiap program studi. Manfaat dari sistem ini adalah dapat mengatasi masalah yang terjadi di UNJ dalam memberikan usulan mahasiswa mana yang akan diterima dalam sebuah program studi.

Penelitian lainnya yang dilakukan oleh [3] adalah membuat sistem pendukung keputusan seleksi penerimaan mahasiswa baru jalur beasiswa dengan metode topsis (*technique for order preference by similarity to ideal solution*). Sistem ini menggunakan metode pengembangan perangkat lunak SDLC (*System Development Life Cycle*) dan implementasi sistem dibangun menggunakan bahasa pemrograman *Delphi* 7 dan pengolahan data menggunakan *DBMS SQL Server* 2012 serta metode perhitungannya menggunakan metode *Technique for Order Preference by Similarity to Ideal Solution* (TOPSIS). Sistem mampu menangani proses seleksi beasiswa dengan tujuh kriteria yaitu akreditasi sekolah, kondisi keluarga, nilai raport, prestasi, organisasi, ujian tertulis, dan wawancara. Serta memberikan kemudahan bagi tim penyeleksi untuk menentukan calon mahasiswa yang berhak mendapatkan beasiswa.

Penelitian selanjutnya yang dilakukan oleh [4] adalah membuat sistem pendukung keputusan penerimaan mahasiswa baru melalui jalur beasiswa bidikmisi di Politeknik Negeri Lhokseumawe menggunakan metode topsis berbasis *web*. Sistem mampu memberikan hasil keputusan yang tepat berdasarkan kriteria yang telah ditetapkan. Adapun kriteria tersebut yaitu kelengkapan berkas, pendapatan orang tua, jumlah tanggungan, orang tua, nilai UN, dan pendidikan. Selain itu, sistem ini

dapat memberikan kemudahan untuk penyeleksian beasiswa diseluruh Indonesia. Metode untuk menentukan hasil seleksi adalah metode *Technique For Others Reference by Similarity to Ideal Solution* (TOPSIS) Sistem ini berbasis *web* yang memudahkan pengguna untuk mengakses dimanapun menggunakan internet.

Penelitian berikutnya yang dilakukan oleh [5] adalah membuat sistem pendukung keputusan menggunakan metode topsis dalam memilih kepala departemen pada kantor balai wilayah Sungai Sumatera II Medan. Perancangan dan pembuatan aplikasi sistem menggunakan bahasa pemograman *Visual Basic, Microsoft Access* 2007, dan metode TOPSIS (*Technique for Order of Preference by Similarity to Ideal Solution*) sebagai metode dalam pembuatan sistem pendukung keputusan ini. Serta kriteria yang digunakan yaitu pendidikan, hasil test, *performance* dan *productivity*. Aplikasi sistem ini bertujuan untuk memudahkan dalam hal pengambilan keputusan pemilihan calon kepala departemen.

Penelitian serupa yang dilakukan oleh [6] adalah mengembangkan sistem pendukung keputusan penerimaan mahasiswa baru pada Lembaga Pendidikan dan Pelatihan Perhotelan Budi Luhur (LP3BL) dengan metode topsis. Sistem ini bertujuan untuk membantu dalam menentukan keputusan seleksi yang tepat untuk penerimaan mahasiswa baru pada Lembaga Pendidikan dan Pelatihan Perhotelan Budi Luhur (LP3BL). Aplikasi sistem ini berbasis desktop, yang dibangun menggunakan bahasa pemrograman *Visual Studio* 2013, *SQL Server* 2012, dan metode perhitungan yang digunakan dalam sistem yaitu metode *Technique For Order Preferences By Similarity To Ideal Solution* (TOPSIS). Serta kriteria yang digunakan yaitu tes fisik, tes tulis bahasa inggris, tes *listening*, dan tes *speaking*.

Pada penelitian ini, penulis bermaksud membangun sebuah sistem pendukung keputusan untuk menentukan penerimaan mahasiswa baru jalur mandiri di Politeknik Negeri Cilacap. Sistem dibangun menggunakan metode TOPSIS. Pada sistem ini, peneliti dalam merekomendasikan calon mahasiswa yang diterima berdasarkan kriteria yang sudah ditentukan. Penelitian yang dilakukan memiliki perbedaan dengan penelitian-penelitian sebelumnya. Perbedaan tersebut diantaranya pada setiap mata pelajaran yang diujikan diberikan bobot yang ditentukan dan diinput oleh masing masing jurusan. Contohnya seperti pada jurusan Teknik Mesin mata pelajaran fisika sangat diperlukan, maka bobot untuk mata pelajaran fisika lebih besar

dibandingkan mata pelajaran yang lain dan juga pada jurusan Teknik Pengendalian Pencemaran Lingkungan mata pelajaran kimia sangat diperlukan, maka bobot untuk mata pelajaran kimia lebih besar. Metode pengembangan perangkat lunak yang dipakai adalah metode *User Centered Design*. Sistem dibuat dengan menggunakan bahasa pemrograman PHP dan *database mysql*.

2.2 Landasan Teori

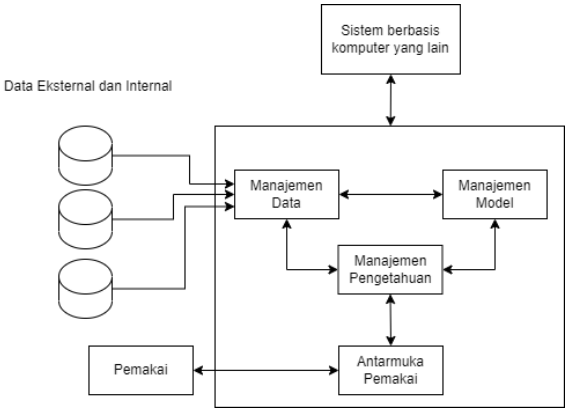
2.2.1 Sistem Pendukung Keputusan

Sistem pendukung keputusan adalah suatu sistem informasi interaktif yang menyediakan informasi, pemodelan, dan pemanipulasi data. Sistem itu digunakan untuk membantu pengambilan keputusan dalam situasi yang semiterstruktur dan situasi tidak terstruktur, dimana tak seorang pun tahu secara pasti bagaiman keputusan seharusnya dibuat [7].

Tujuan sistem pendukung keputusan [7]:

1. Membantu manajer dalam pengambilan keputusan atas masalah semi terstruktur.
2. Memberikan dukungan atas pertimbangan manajer dan bukannya di maksudkan untuk menggantikan fungsi manajer.
3. Meningkatkan efektivitas keputusan yang di ambil manajer lebih daripada perbaikan efisiensinya.
4. Kecepatan komputasi. Komputer memungkinkan para pengambil keputusan untuk melakukan banyak komputasi secara cepat dengan biaya yang rendah.
5. Peningkatan produktivitas.
6. Dukungan kualitas.
7. Berdaya saing.
8. Mengatasi keterbatasan kognitif dalam pemrosesan dan penyimpanan.

Model konseptual sistem pendukung keputusan dengan *database, user,* maupun sistem berbasis komputer yang lain digambarkan seperti pada Gambar 2.1 [8]



Gambar 2. 1 Model Konseptual SPK

2.2.2 TOPSIS *(Technique for Order Preference by Similiarity to Ideal Solution)*

TOPSIS *(Technique for Orders Preference by Similarity to Ideal Solution)* adalah salah satu metode pengambilan keputusan multikriteria. TOPSIS menggunakan prinsip bahwa alternatif yang terpilih harus mempunyai jarak terdekat dari solusi ideal positif dan terjauh dari solusi ideal negatif dari sudut pandang geometris dengan menggunakan jarak *Euclidean* untuk menentukan kedekatan relatif dari suatu alternatif dengan solusi optimal. Solusi ideal positif didefinisikan sebagai jumlah dari seluruh nilai terbaik yang dapat dicapai untuk setiap atribut, sedangkan solusi negatif ideal terdiri dari seluruh nilai terburuk yang dicapai untuk setiap atribut.

TOPSIS mempertimbangkan keduanya, jarak terhadap solusi ideal positif dan jarak terhadap solusi ideal negatif dengan mengambil kedekatan relatif terhadap solusi ideal positif. Berdasarkan perbandingan terhadap jarak relatifnya, susunan prioritas alternatif bisa dicapai. [4]

Langkah-langkah dalam metode TOPSIS, adalah [3]:

1. Membangun *normalized decision matrix*

Elemen Rij hasil dari normalisasi *decision matrix R* dengan metode *Euclidean length of a vector* adalah:

$$R_{ij} = \frac{X_{ij}}{\sqrt{\sum_{i=1}^{m} X_{ij}^2}}$$

Persamaan 2. 1

dengan i = 1,2,3, …m; dan j = 1,2,3 … n

2. Membangun *weighted normalized decision matrix*

Dengan bobot W=(w1,w2…,wn), maka normalisasi bobot matriks V adalah:

$$V=\begin{bmatrix} w_1 r_{22} & x_{12} & \cdots & w_{nr1n} \\ w_1 r_{21} & x_{22} & \cdots & w_{nr2n} \\ . \quad . & . & \cdots & . \\ w_{1rm1} & w_{2rm2} & \cdots & w_{nrmn} \end{bmatrix}$$ **Persamaan 2. 2**

3. Menentukan solusi ideal positif dan solusi ideal negatif

Solusi ideal positif A^+ dan solusi ideal negatif A^- dapat ditentukan berdasarkan rating bobot ternormalisasi (*Yij*) sebagai:

$$Y_{ij} = W_i R_{ij}$$ **Persamaan 2. 3**

dengan i = 1,2,3, …m; dan j = 1,2,3 … n

Solusi ideal positif (A^+) dihitung berdasarkan:

$A^+ = (y_1^+, y_2^+, y_3^+, \dots, yj^+)$ **Persamaan 2. 4**

Solusi ideal negatif (A^-) dihitung berdasarkan:

$A^- = (y_1^-, y_2^-, y_3^-, \dots, yj^-)$ **Persamaan 2. 5**

Dengan

$$y_j^+ \begin{cases} \max\limits_i y_{ij}; \textit{ jika j adalah atribut keuntungan} \\ \min\limits_j y_{ij}; \textit{ jika j adalah atribut biaya} \end{cases}$$

$$y_j^- \begin{cases} \min\limits_i y_{ij}; \textit{ jika j adalah atribut keuntungan} \\ \max\limits_j y_{ij}; \textit{ jika j adalah atribut biaya} \end{cases}$$

j=1,2,…n

4. Menghitung separasi Di*

Jarak (dalam pandangan *Euclidean*) alternatif dari solusi positif-ideal didefinisikan sebagai:

$D_i^+ = \sqrt{\sum_{j=1}^{n}(y_{ij} - y_i^+)^2}$ **Persamaan 2. 6**

dengan i = 1,2,3, …m; dan j = 1,2,3 … n

Dan jarak terhadap solusi negatif-ideal didefinisikan sebagai:

$D_i^- = \sqrt{\sum_{j=1}^{n}(y_{ij} - y_i^-)^2}$ **Persamaan 2. 7**

dengan i = 1,2,3, …m; dan j = 1,2,3 … n

5. Menghitung kedekatan relatif terhadap solusi ideal

$V = \frac{D_i^-}{D_i^- + D_i^+}$ **Persamaan 2. 8**

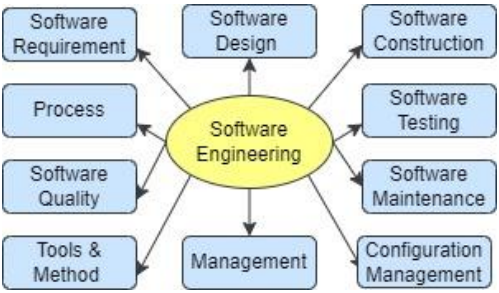
dengan i = 1,2,3, …m; dan j = 1,2,3 … n

2.2.3 Rekayasa Perangkat Lunak

Perangkat lunak *(software)* adalah program computer yang terasosiasi dengan dokumentasi perangkat lunak seperti dokumentasi kebutuhan, model desain, dan cara penggunaan *(user manual)*. Rekayasa perangkat lunak *(software engineering)* atau RPL merupakan pembangunan dengan menggunakan prinsip atau konsep rekayasa dengan tujuan menghasilkan perangkat lunak yang bernilai ekonomi yang dipercaya dan bekerja secara efisien menggunakan mesin. Rekayasa Perangkat Lunak lebih fokus pada bagaimana membuat perangkat lunak yang memenuhi kriteria berikut [9]:

1. Dapat terus dipelihara setelah perangkat lunak selesai dibuat seiring berkembangnya teknologi dan lingkungan *(maintainability)*.
2. Dapat diandalkan dengan proses bisnis yang dijalankan dan perubahan yang terjadi *(dependability dan robust)*.
3. Efisien dari segi sumber daya dan penggunaan.
4. Kemampuan untuk dipakai sesuai dengan kebutuhan *(usability)*.

Ruang lingkup rekayasa lunak menurut Abran dapat digambarkan sebagai berikut [10]:



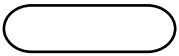
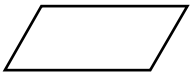
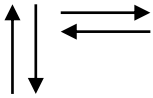
Gambar 2. 2 Ruang Lingkup RPL

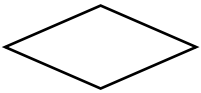
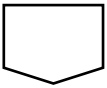
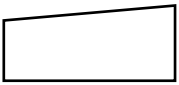
1. *Software requirement*, berhubungan dengan spesifikasi kebutuhan dan persyaratan perangkat lunak.
2. *Software design* mencakup proses penentuan arsitektur, komponen, antarmuka, dan karakteristik lain dari perangkat lunak.
3. *Software construction*, berhubungan dengan detail pengembangan perangkat lunak, termasuk algoritma, pengkodean, pengujian, dan pencarian kesalahan.
4. *Software testing*, meliputi pengujian pada keseluruhan perilaku perangkat lunak.
5. *Software maintenance*, mencakup upaya-upaya perawatan ketika perangkat lunak telah dioperasikan.
6. *Software configuration management*, berhubungan dengan usaha perubahan konfigurasi perangkat lunak untuk memenuhi kebutuhan tertentu.
7. *Software engineering management*, berkaitan dengan pengelolaan dan pengukuran RPL, termasuk perencanaan proyek perangkat lunak.
8. *Software engineering tools and methods*, mencakup kajian teoritis tentang alat bantu dan metode RPL.
9. *Software engineering process*, berhubungan dengan definisi, implementasi, pengukuran, pengelolaan, perubahan dan perbaikan proses RPL.
10. *Software quality*, menitikberatkan pada kualitas dan daur hidup perangkat lunak.

2.2.4 Flowchart

Flowchart adalah penggambaran secara grafik dari langkah-langkah dan urut-urutan prosedur dari suatu program. *Flowchart* biasanya mempermudah penyelesaian suatu masalah yang perlu dipelajari dan dievaluasi lebih lanjut [11]. Berikut Tabel 2.1 adalah simbol-simbol yang ada ada *flowchart* [11]:

Tabel 2. 1 Simbol-Simbol *Flowchart*

No	Simbol	Nama	Keterangan
1		*Terminal*	Memulai dan mengakhiri suatu program
2		*Input/output*	Memasukkan data maupun menunjukkan hasil dari suatu *process* tanpa tergantung dengan jenis peralatannya.
3		*Process*	Menunjukkan pengolahan yang akan dilakukan oleh computer.
4		*Connector*	Menyatakan sambungan dari proses yang satu ke proses berikutnya di halaman yang sama.
5		*Flow*	Menghubungkan antara simbol satu dengan simbol yang lain atau menyatakan jalannya arus dalam suatu proses. Simbol arus ini sering disebut juga dengan

			connecting line.
6		*Decision*	Memilih proses yang akan dilakukan berdasarkan kondisi tertentu.
7		*Predefined Process*	Menunjukkan pelaksanaan suatu bagian prosedur (sub-proses).
8		*Preparation*	Proses *inisialisasi* atau pemberian harga awal.
9		*Document*	Menyatakan masukan yang berasal dari dokumen dan keluaran yang berupa dokumen
10		*Off Page Connector*	Menyatakan sambungan dari proses yang satu ke proses berikutnya di halaman yang berbeda.
11		*Manual Operation*	Menunjukkan proses yang tidak dilakukan oleh komputer.
12		*Manual Input*	Memasukkan data secara manual menggunakan *online* keyboard.

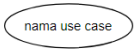
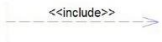
2.2.5 UML *(Unified Modeling Language)*

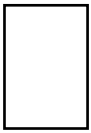
UML *(Unified Modeling Language)* merupakan standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun menggunakan teknik pemrograman berorientasi objek. UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak [9].

1. *Use Case Diagram*

Use case atau diagram *use case* merupakan pemodelan untuk kelakuan *(behavior)* sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih actor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Berikut pada Tabel 2.2 adalah simbol-simbol yang ada pada diagram *use case* [9]:

Tabel 2. 2 Simbol-Simbol *Use Case*

No	Simbol	Nama	Keterangan
1		*Use case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suat hasil yang terukur bagi suatu aktor.
2		*Actor*/aktor	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
3		*Generalization*	Hubungan dimana objek anak *(descendent)* berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk *(ancestor)*.
4		*Include*	Menspesifikasikan bahwa *use case* sumber secara *eksplisit*.

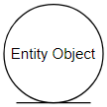
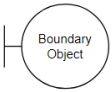
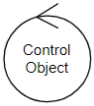
5		*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang dberikan.
6		*Association*	Menghubungkan antara objek satu dengan objek lainya.
7		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.

2. Sequence Diagram

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu, untuk menggambar *sequence diagram* maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat *sequence diagram* juga dibutuhkan untuk melihat skenario yang pada *use case*. Berikut pada Tabel 2.3 adalah simbol-simbol yang ada pada *sequence diagram* [9]:

Tabel 2. 3 Simbol-Simbol *Sequence Diagram*

No	Simbol	Nama	Keterangan
1		*Actor*	Menggambarkan *user* atau pengguna.

2		*Entity Class*	Menggambarkan hubungan yang akan dilakukan.
3		*Boundary Class*	Menggambarkan sebuah form
4		*Control Class*	Menghubungkan antara *boundary* dengan tabel.
5		*Lifeline*	Menggambarkan hubungan kegiatan yang akan dilakukan.
6		*Massage*	Menggambarkan pengiriman pesan.

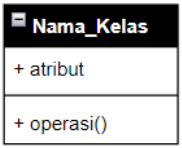
3. Class Diagram

Class diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

a. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas.
b. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Berikut pada Tabel 2.4 adalah simbol-simbol yang ada pada *class diagram* [9] :

Tabel 2. 4 Simbol-Simbol *Class Diagram*

No	Simbol	Nama	Keterangan
1		Kelas	Kelas pada struktur sistem.
2		Antarmuka/ *Interface*	Sama dengan konsep *interface* dalam pemrograman berorientasi objek.
3		*Association*	Relasi antarkelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity.*
4		*Directed Association/* Asosiasi Berarah	Relasi antarkelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity.*
5		Generalisasi	Relasi antarkelas dengan makna generalisasi-spesialisasi (umum khusus).
6		*Dependency/* Kebergantungan	Relasi antarkelas

			dengan makna kebergantungan antarkelas.
7		*Aggregation/* Agregasi	Relasi antarkelas dengan makna semua-bagian *(whole-part).*

2.2.6 Basis Data

Basis data merupakan koleksi dari data yang terorganisasi dengan cara sedemikian rupa sehingga data tersebut mudah disimpan dan dimanipulasi. Di samping berisi atau menyimpan data, setiap basis data juga mengndung/menyimpan definisi struktur [12].

1. **DBMS *(Database Management System)***

DBMS *(Database Management System)* atau dalam bahasa Indonesia sering disebut sebagai Sistem Manajemen Basis Data adalah suatu sistem aplikasi yang digunakan untuk menyimpan, mengelola, dan menampilkan data[9]. Dalam penggunaan DBMS, dibutuhkan komponen-komponen antara lain [13]:

a. *Query Processor,* komponen yang mengubah bentuk *query* dalam bentuk instruksi ke dalam *database manager*.
b. *Database Manager*, menerima *query*, mnguji eksternal dan konseptual untuk menentukan apakah *record-record* tersebut dibutuhkan untuk memenuhi permintaan kemudian hari dari *database manager* dengan memanggil *file manager* untuk menyelesaikan permintaan.
c. *File Manager,* memanipulasi penyimpanan *file* dan mengatur alokasi ruang penyimpanan *disk.*
d. *DML Processor,* modul yang mengubah perintah DML yang ditempelkan ke dalam program aplikasi dalam bentuk fungsi-fungsi.
e. *DDL Compiler*, mengubah statement DDL, menjadi kumpulan Tabel atau *file* yang berisi data *dictionary* atau meta data.
f. *Dictionary manajer,* mengatur akses dan memelihara data *dictionary.*

Salah satu software yang tergolong ke dalam DBMS adalah *MySQL*. *MySQL* merupakan salah satu *Relational Database Management System (RDBMS)* yang saat ini sedang banyak diminati

untuk menyimpan, mengatur, dan mengelola data pada aplikasi tersebut. Beberapa kelebihan *MySQL* dibandingkan dengan RDBMS lain adalah mudah, simple, gratis, stabil, dan portable (dapat diterapkan pada beberapa sistem operasi yang berbeda) [14].

2. SQL *(Structured Query Language)*

SQL *(Structured Query Language)* adalah bahasa yang digunakan untuk mengelola data pada RDBMS. SQL awalnya dikembangkan berdasarkan teori aljabar relasional dan kalkulus [9]. Secara umum SQL terdiri dari 2 (dua) bahasa yaitu *Data Definition Language*(DDL) dan *Data Manipulation Language*(DML) [12].

a. DDL (*Data Definition Language)*

DDL *(Data Definition Language)* merupakan suatu perintah yang berfungsi untuk mendefinisikan atribut-atribut basis data, Tabel, atribut serta hubungan antar Tabel. DDL berfungsi lebih ke dalam memanipulasi struktur dari *database*. DDL digunakan untuk membuat Tabel atau menghapus Tabel, membuat *key* atau *indeks*, membuat relasi antar Tabel. Berikut *statement* atau sintaks yang ada di DDL [12]:

1. *Create table*, bertugas untuk membuat Tabel
2. *Create index*, bertugas untuk membuat suatu indeks dalam Tabel
3. *Drop table*, bertugas untuk menghapus suatu Tabel
4. *Drop index*, bertugas untuk menghapus suatu indeks dalam Tabel
5. *Alter table*, bertugas untuk merubah struktur suatu Tabel

b.* DML *(Data Manipulation Language)

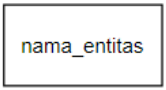
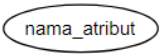
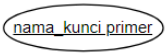
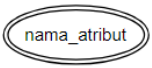
DML *(Data Manipulation Language)* merupakan kelompok perintah yang berfungsi untuk memanipulasi data dalam basis data. DML bisa digunakan untuk melakukan proses *insert, update*, atau *delete* ke dalam suatu *database*. Berikut sintaks yang ada dalam DML [12]:

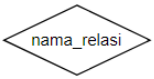
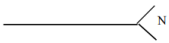
1. *Insert*, bertugas untuk menambahkan data ke dalam suatu Tabel dalam *database*
2. *Update*, bertugas untuk *update* (merubah) data dalam suatu Tabel pada *database*
3. *Select*, bertugas untuk mengakses data dari suatu Tabel dalam *database*
4. *Delete*, bertugas untuk menghapus data dari suatu Tabel dalam *database*

3. ERD *(Entity Relationship Diagram)*

ERD *(Entity Relationship Diagram)* merupakan suatu model jaringan yang menggunakan susunan data yang disimpan pada sistem secara abstrak. ERD juga menggambarkan hubungan antara satu entitas yang memiliki sejumlah atribut dengan entitas lain dalam suatu sistem yang terintegrasi. ERD digunakan oleh perancang sistem untuk memodelkan data yang nantinya akan dikembangkan menjadi basis data *(database)* [12]. Berikut pada Tabel 2.5 adalah simbol-simbol yang digunakan pada ERD dengan notasi Chen [9]:

Tabel 2. 5 Simbol-Simbol ERD

No	Simbol	Nama	Keterangan
1		Entitas/*Entity*	Entitas merupakan data inti yang akan disimpan.
2		Atribut	*Field* atau kolom data yang butuh disimpan dalam suatu entitas.
3		Atribut kunci primer	*Field* atau kolom data yang butuh disimpan dalam suatu entitas dan digunakan sebagai kunci akses *record* yang diinginkan, biasanya berupa id.
4		Atribut multinilai/ *multivalue*	*Field* atau kolom data yang butuh disimpan dalam suatu entitas yang dapat memiliki lebih dari satu.

5		Relasi	Relasi yang menghubungkan antar entitas, biasanya diawali dengan kata kerja
6		Asosiasi / *association*	Penghubung antara relasi dan entitas dimana di kedua ujungnya memiliki multiplicity kemungkinan jumlah pemakaian. Kemungkinan jumlah maksimum keterhubungan antara entitas satu dengan entitas yang lain disebut dengan kardinalitas. Misalkan ada kardinalitas 1 ke N atau sering disebut dengan one to many yang menghubungkan entitas A dan entitas B.

2.2.7 Penerimaan Mahasiswa Baru Jalur Mandiri Politeknik Negeri Cilacap

Peneriman mahasiswa baru adalah salah satu kegiatan utama perguruan tinggi yang selalu dilaksanakan setiap tahunnya, salah satunya di Politeknik Negeri Cilacap. Politeknik Negeri Cilacap merupakan satu-satunya perguruan tinggi negeri di Cilacap yang memberikan kesempatan untuk putra daerah mengembangkan *skill* dan jiwa wirausaha *(technopreneurship)* di kampus negeri. Ada 5 jalur masuk dalam penerimaan mahasiswa baru diantaranya untuk D3 SNMPN dan SBMPN, D4 SNMPTN dan SBMPTN, dan yang terakhir jalur Mandiri.

Penerimaan mahasiswa baru melalui jalur mandiri merupakan penerimaan mahasiswa dengan jalur ujian masuk perguruan tinggi yang dilaksanakan secara mandiri oleh Politeknik Negeri Cilacap. Dalam jalur mandiri terdapat 3 macam tes yaitu tes ujian tertulis dengan 5 mata pelajaran (matematika, fisika, kimia, bahasa Indonesia, dan bahasa inggris), tes buta warna, dan yang terakhir tes ujian tertulis. Di jalur mandiri terdapat perbedaan dengan jalur masuk lainnya yaitu di jalur mandiri proses seleksi dilaksanakan oleh panitia PMB PNC selain itu juga bagi calon mahasiswa baru yang lolos akan dikenai SPI (sumbangan pengembangan institusi) dimana di jalur masuk yang lain tidak dikenai pembayaran SPI.